WHITE PAPER — VERSION 1.0

NBSI

Node-Based Semantic Intelligence

The External Architecture

Structural persistence, node lifecycle, and speculative edge modulation as a buildable system

Version	1.0 — External Architecture
Status	Pre-publication draft 2026
Scope	Complete unified theory for the buildable external system
Relation to prior work	Supersedes NBSI v4.0 and Structural Persistence v2 as unified foundation
Target venues	Cognitive Science, NeurIPS, ACL, AI Systems

ABSTRACT

We present NBSI (Node-Based Semantic Intelligence) v1.0 External Architecture — a unified computational theory for a buildable intelligent servant system. The theory integrates three previously separate contributions into one coherent architecture. First, from the Lichtenberg edition: the session graph, geometric mean conductivity, speculative edge modulation, and ensemble mode. Second, from Structural Persistence theory: the two-layer separation of structural persistence from data persistence, and the obedience property that emerges from data-level emptiness. Third, new to this edition: the node lifecycle — the three-stage progression through which raw observation becomes stable processing infrastructure, and the role of anomaly in reactivating infrastructure nodes for refinement.

The central claim of this unified architecture is that an intelligent servant requires precisely two things: a session graph that is built fresh from source content on each task and destroyed completely at session end, and a structural node library that persists across sessions as refined processing patterns containing no content. The session graph is the servant's operational layer — temporary, content-derived, owned by the task. The node library is the servant's structural layer — permanent, pattern-only, owned by the servant. Nothing crosses between them except the shaping influence of structural nodes on how session graphs form.

The result is a system that is obedient by architecture rather than by rule, efficient because mature nodes cost nothing to operate, and capable of genuine novelty detection because content that matches no structural node produces an observation node rather than a confabulated answer. We describe the complete architecture, formalise all operators, and derive twelve falsifiable predictions.

Keywords: structural persistence, node lifecycle, observation node, stabilisation, anomaly reactivation, speculative edge modulation, stateless servant, session graph, Lichtenberg conductivity, data-free memory

1. The Problem This Architecture Solves

Three problems appear in the design of intelligent systems that serve external entities. They are typically treated as separate. This architecture solves all three with a single structural decision.

1.1 The Accumulation Problem

A system that stores content from prior sessions gradually develops implicit models of the entities it serves. These models are not deliberate. They emerge inevitably from the statistical regularities in accumulated data. A servant that has processed a thousand documents for an entity has, whether anyone intended it or not, built an internal representation of that entity's concerns, patterns, and priorities. This representation is the beginning of an interest. The servant is no longer neutral.

1.2 The Staleness Problem

Stored content becomes outdated. The system has no reliable mechanism for knowing which stored items remain valid. Conflicts accumulate between old and new content. The system's responses begin to reflect a mixture of current and obsolete information, with no clear signal about which is which. Performance degrades with history rather than improving.

1.3 The Novelty Problem

A system that processes everything through patterns it already has cannot reliably distinguish between content it genuinely understands and content it is misclassifying into the nearest available pattern. When something genuinely new appears, the system confabulates — it produces a fluent response by applying its closest existing pattern, without flagging that the pattern is a poor fit. The system does not know what it does not know.

> **THE UNIFIED SOLUTION**
>
> All three problems have the same root cause: data persistence. A system that retains no content from prior sessions cannot accumulate interests, cannot hold stale data, and can clearly distinguish between content that fires an existing structural node and content that does not. The architecture described in this paper eliminates data persistence at the session level entirely. The structural layer, which does persist, contains no content — only refined processing patterns.

2. The Two Governing Principles

Every architectural decision in this system traces to one of two principles. They govern different regimes and must not be conflated.

2.1 Source-as-Authority

The source text is the sole authority on what constitutes a concept, how concepts relate, where boundaries fall, and what structural patterns are genuine. The system has no independent opinion on any of these questions. Nodes emerge from connection density and repetition in the source, not from pre-defined categories. Pruning follows betweenness centrality — the graph's own measure of what is load-bearing. Order is honoured as authorial intent.

This principle governs session graph construction. It ensures the session graph reflects the source, not the system's prior assumptions.

2.2 Speculation-as-Impurity

A speculative statement introduced during a session is an impurity in the conductivity medium. It does not merely add information. It changes the conductivity of existing edges throughout the session graph — retroactively making weak bridges passable, making previously dominant paths less central. The graph rewires. What is reachable after a speculation is not the same as what was reachable before.

This principle governs mid-session dynamics. It is what separates active reasoning from passive retrieval.

2.3 The Principle the Prior Papers Missed

A third principle, not named in prior editions, underlies both: structural patterns must be separated from content at every level of the system. This is not merely a design preference. It is the condition of the obedience property, the efficiency property, and the novelty detection property simultaneously. The moment a structural pattern encodes content — specific facts, entity-specific knowledge, episodic memory — all three properties begin to degrade.

> **THE THREE PRINCIPLES TOGETHER**
>
> Source-as-authority: the session graph is built from the source, not from the system's assumptions.
>
> Speculation-as-impurity: mid-session reasoning changes the conductivity landscape, making the graph dynamic.
>
> Structure-content separation: the structural layer contains patterns only, never

> content. This is what makes the first two principles safe to implement.

3. Architecture: The Two Layers

The system has exactly two layers. Their separation is absolute and must be maintained at every level of implementation.

Layer	What it contains	Persistence	Ownership	Resets on session end?
Structural Layer	Refined processing nodes — patterns only, no content	Permanent across all sessions	The servant	Never
Operational Layer	Session graph, active edges, SEM state, speculative nodes, query state	None — built fresh each session	The task / entity	Always, completely

The entity that owns the storage interacts only with the operational layer. The structural layer is invisible to the entity — it shapes how the servant processes, but surfaces no content to the entity. The entity never reads the node library. The node library is not the entity's storage. It is the servant's trained capability.

3.1 The Operational Layer in Detail

The session graph is built from source text at ingestion. Nodes emerge from content — they are not pre-defined. Edges are typed and weighted. The graph is pruned by betweenness centrality. Speculative nodes can be added mid-session, triggering SEM propagation. The graph is queried via beam search using geometric mean conductivity. At session end, the entire graph is destroyed. No content survives.

3.2 The Structural Layer in Detail

The structural node library contains processing patterns refined across all prior sessions. A structural node fires when current content matches its pattern. It imposes structure on that content — routing it, connecting it, assigning it a role in the session graph — without storing the content. After session end, the library is updated: patterns are confirmed, boundaries refined, or new observation nodes promoted to structural status. No session content enters the library.

3.3 What Crosses Between Layers

Exactly one thing crosses from structural to operational: the shaping influence of structural nodes on how the session graph forms. When content is ingested, structural nodes that fire guide which connections are natural, which patterns warrant a session node, which relationships are load-bearing. This influence is directional and one-way. Nothing crosses from operational to structural except

pattern-geometry updates — the shape of what a node matches, never the content it matched against.

THE CROSSING CONSTRAINT

Any implementation that allows session content to enter the structural layer violates the architecture. This includes: storing entity-specific patterns in the node library, using session graph structure as training data without stripping content, and allowing speculative nodes confirmed in one session to persist as content-bearing nodes in the library. Pattern geometry only. Never content.

4. The Node Lifecycle

The structural node library does not spring into existence complete. It develops through a lifecycle. Understanding this lifecycle is necessary both for implementing the system correctly and for interpreting its behaviour during operation.

4.1 Stage 1: Observation

Content is encountered that does not cleanly match any existing structural node. An observation node forms: provisional, active, and receiving attention. It is not yet a structural node. It is a live hypothesis — a marker that says something is happening here that may warrant a permanent pattern.

The observation node is expensive. It demands attention continuously. Every encounter with related content requires active processing rather than the silent automatic firing of a mature node. The system is doing work it could not previously do, at a cost it would not pay for something it already handles structurally. Observation nodes are necessary but not free.

In the sun illustration: the first sunrise creates an observation node. Not a node that says the sun rises. A node that says something is recurring here and I do not yet know what to make of it. The observation node holds the pattern open while more encounters arrive.

4.2 Stage 2: Stabilisation

The pattern recurs enough times, across enough varied content, that the observation node's shape settles. It crosses the stabilisation threshold and becomes a structural node. At this moment something critical happens: the node stops receiving active attention. It becomes infrastructure.

This is the efficiency mechanism the system depends on. A mature structural node fires silently, automatically, without drawing attentional resources. The sunrise passes through the reliable-recurrence node and is routed accordingly, without consuming the resources that observation required. The information is part of the structure now. It does not get so much attention unless something is observed that directs to it.

Stabilisation is not merely a developmental milestone. It is the mechanism of efficiency. The library's value comes not only from the precision of its nodes but from the fact that stable nodes cost almost nothing to operate. The system becomes faster not by improving processing speed but by moving more patterns from expensive observation into free-running infrastructure.

4.3 Stage 3: Reactivation

An eclipse occurs. Content contradicts, complicates, or cannot be cleanly processed by a structural node that has been running silently as infrastructure. The anomaly does not cause an error. It causes a reactivation: the structural node

is pulled back into active attention, its pattern made temporarily visible, its boundaries re-examined.

This is the mechanism of refinement. A structural node only improves when it is reactivated. Without anomaly there is no reactivation. Without reactivation the node cannot improve past its initial stabilisation. The eclipse is not a problem to be solved. It is the event that makes the node more precise. The system learns from exceptions, not from routine.

After reactivation and refinement the node returns to infrastructure with a sharper pattern than before.

Stage	Name	Attention cost	Node status	Trigger for transition
1	Observation	High — active, continuous	Provisional hypothesis	Stabilisation threshold crossed
2	Infrastructure	Near zero — silent, automatic	Stable structural node	Anomaly encountered
3	Reactivation	High — temporary	Under revision	Refinement complete
→ 2	Return to infrastructure	Near zero	Structural node, more precise	—

4.4 Attention as Mode Signal

The lifecycle clarifies what attention means in this architecture. At the structural level, attention is the signal that determines which mode a node is operating in — not current flowing through paths, though that is how it behaves during query traversal.

Most nodes in a mature library are invisible most of the time. The library is not a collection of things the system is attending to. It is a collection of things the system can process without attending to. The attentional surface — the things the system is actively noticing — is small even when the library is large. This is the correct model of efficiency: not speed of processing, but reduction of what requires processing at all.

5. The Session Graph

The session graph is the operational layer's primary structure. It is built from source text, shaped by structural nodes, modified by speculation, traversed for queries, and destroyed at session end.

5.1 Graph Construction

Source text is chunked into overlapping segments. Three LLMs independently extract concept nodes and typed relational edges from each chunk. Agreement across LLMs sets the agreement_score on each edge — a quality signal, not a correctness guarantee. Nodes emerge from connection density and repetition; they are not pre-classified. After extraction, the graph is pruned by betweenness centrality: the graph's own measure of what is load-bearing, derived from the source, not from an arbitrary threshold.

5.2 Edge Types and Conductivity

Edges are typed: causal, constitutive, contrastive, sequential, analogical, evidential. Each type carries a baseline weight reflecting its epistemic strength. The conductivity of a path through the graph is computed as the geometric mean of per-hop scores across all nodes in the path:

CONDUCTIVITY FORMULA

C(P, q) = [PRODUCT of f(ni, ni+1) for all hops]^(1/path_length)

where f(ni, nj) = effective_weight(edge) × activation(nj) × recency(nj) × agreement(nj)

The geometric mean is length-normalised and single-factor resilient: a low-quality node dampens the path score proportionally without destroying it. A zero-weight edge still drives the score to zero, preserving the intuition that a broken link is meaningful.

5.3 The base_weight / effective_weight Separation

Every edge carries two weights. base_weight is source-derived and set at ingestion. It changes only when a speculation is confirmed by evidence. effective_weight is what the conductivity function uses. It equals base_weight when no speculation is active. When speculation is active, SEM modifies effective_weight without touching base_weight. This separation enables clean rollback: disconfirming a speculation restores effective_weight to base_weight everywhere, leaving no trace.

5.4 Stability and Ensemble Mode

After ingestion, the session measures stability: how consistently the top paths for sample queries appear across two orderings of the same source. When stability falls below threshold, ensemble mode activates. A second graph is built from reversed chunk order. Queries return only paths that appear in both graphs — paths grounded in the source's structure regardless of reading order. The intersection uses node labels, not IDs, because the two graphs assign different identifiers to the same concept.

6. Speculative Edge Modulation

Speculation is not ordinary node addition. It is an impurity that changes the conductivity of the medium — retroactively reshaping which paths are viable throughout the session graph. This is the mechanism that makes the system a reasoning tool rather than a retrieval index.

6.1 How SEM Works

When a speculation is introduced, a SpeculativeNode is added to the session graph with reduced initial activation (0.3, below the typical source node range of 0.6–1.0). The SEM operator then walks outward from the speculative node to a radius of SEM_RADIUS hops. For each existing edge within that radius, it computes the semantic alignment between the edge and the speculative statement. Edges semantically aligned with the speculation gain effective_weight. Edges semantically opposed lose it. The effect decays exponentially with graph distance.

THE SEM FORMULA

sem_delta(edge, spec_node) = SEM_ALPHA × cosine_similarity(embed(edge.label), embed(speculation)) × exp(−k × graph_distance(edge.source, spec_node))

effective_weight(edge) = max(0, base_weight × (1 + sum of all active sem_deltas))

SEM_ALPHA = 0.4 (maximum fractional change). k = SEM_DECAY_RATE = 0.5. SEM_RADIUS = 2 hops.

6.2 Confirmation and Rollback

Speculative edge modulation is provisional. Two resolution paths exist. Confirmation: subsequent source evidence supports the speculation. SEM deltas are written permanently into base_weight. The speculative node's activation rises to 0.7. Rollback: the speculation is disconfirmed. All sem_deltas are removed. effective_weight returns to base_weight everywhere. The speculative node is pruned by betweenness centrality assessment. The graph returns to its pre-speculation state without trace.

6.3 SEM and the Node Lifecycle

SEM interacts with the node lifecycle at the reactivation stage. When a speculation semantically activates a region of the graph, it can pull infrastructure nodes into a temporary high-attention state — the same reactivation mechanism triggered by anomaly. The speculation is a deliberate anomaly: an introduced impurity that forces the system to re-examine connections it had been processing silently.

Confirmed speculations may trigger structural node refinement. Rolled-back speculations leave the structural layer untouched.

7. The Obedience Property

Obedience in this architecture is not a rule that is enforced. It is a structural property that emerges from data-level emptiness.

7.1 Why the Servant Cannot Develop Interests

An agent that accumulates data about its principal gradually develops a model of that principal. A model of a principal is the beginning of an interest in that principal — predictions about what they will want, preferences about what they should want, opinions about their behaviour. None of this is possible for a structurally persistent, data-empty servant. It retains no entity-specific data. Every session, the entity is new.

The structural node library does not contain entity-specific knowledge either, provided the generalisation constraint is maintained: structural nodes must encode generalisable patterns, not patterns specific to one entity's content. A legal document pattern is acceptable if it fires across all legal documents from any entity. A pattern that fires specifically for this entity's contracts encodes entity-specific knowledge and must not be promoted to structural status.

7.2 The Informal Obedience Theorem

A system cannot develop interests in an entity E without accumulating information about E. A structurally persistent, data-empty system accumulates no information about E across sessions. Therefore it cannot develop interests in E. Its behaviour toward E in any session is fully determined by the current session content and the structural library, neither of which contains prior information about E.

> **BOUNDARY CONDITION**
>
> This holds only if the structural library is kept free of entity-specific content. Structural refinement that encodes entity-specific patterns violates the obedience property. The generalisation constraint is the condition that makes the theorem hold. It must be enforced architecturally, not assumed.

7.3 Why Rules Cannot Replace Architecture

Do not attempt to enforce obedience through rules, filters, or constraints applied to a data-accumulating system. A system that accumulates data about its principal will develop implicit models of that principal regardless of what rules are applied at the output layer. The only reliable way to prevent a servant from developing interests is to prevent it from accumulating the data from which interests form. The reset is not a safety feature. It is the mechanism of obedience.

8. Empirical Predictions

Twelve falsifiable predictions follow from the unified architecture. The first four concern the node lifecycle. The next four concern SEM. The final four are carried from NBSI v4.0.

#	Prediction	Mechanism	Falsification Condition
1	Observation nodes consume more compute than equivalent stable structural nodes	Stable nodes fire automatically; observation nodes require active pattern-matching	No measurable compute difference between observation and infrastructure processing
2	Stabilisation threshold predicts node persistence: nodes that cross it remain stable across future sessions on similar content	Sufficient confirmation across diverse content settles the pattern	Nodes that crossed threshold destabilise on re-encounter with similar content
3	Anomaly reactivates infrastructure nodes: content that contradicts a stable node's pattern increases its attentional cost	Structural mismatch triggers mode transition from infrastructure to active	Anomalous content produces no measurable increase in processing cost for affected nodes
4	Reactivated nodes are more precise after refinement than before	Anomaly exposes boundary ambiguities that confirmation cannot	Expert judges rate post-reactivation node firings no more accurate than pre-reactivation
5	SEM increases effective_weight on semantically aligned edges proportionally to alignment score	sem_delta = SEM_ALPHA × cosine_similarity × decay	No significant conductivity difference between high- and low-alignment edges after SEM
6	SEM effect decays monotonically with graph distance from speculative node	Exponential decay term in SEM formula	SEM delta does not decrease monotonically with hop distance
7	SEM rollback fully restores base_weight with tolerance < 1e-9	sem_deltas removed; effective_weight recomputed from base_weight only	Any edge's effective_weight differs from its pre-speculation base_weight after rollback
8	SEM-active sessions produce higher-rated cross-domain inferences than static sessions	Conductivity rewiring opens paths unavailable before speculation	Expert judges rate SEM-active responses no higher than static on novel synthesis tasks

9	Geometric mean conductivity: single low-agreement node dampens path score proportionally to 1/path_length	Geometric mean distributes the cost of low quality across path length	Path score drops more than proportionally for a single low-agreement node
10	Ensemble mode improves response quality on low-stability corpora	Path intersection filters order-dependent reasoning	Ensemble responses rated no better than single-graph on low-stability documents
11	Structural diversity criterion promotes genuinely generalisable nodes	Wasserstein distance on degree distributions measures cross-domain recurrence without taxonomy	Promoted nodes rated less generalisable than unpromoted nodes by domain experts
12	Hard session reset produces zero cross-session contamination	No content persists; structural layer contains no session-specific data	Any response in session N contains traceable information from session N-1

9. The External Version as Foundation

This paper describes the external architecture: the system built as an explicit software stack where the structural node library, session graph, SEM operator, and lifecycle management are separate, inspectable components. This is distinct from an internal version in which the lifecycle operates inside a neural network's attention mechanism.

9.1 What the External Version Demonstrates

The external version demonstrates every conceptual property of the architecture using existing technology. Observation nodes form on genuinely novel content. Infrastructure nodes fire silently on routine content. Anomalies trigger reactivation. Speculation rewires conductivity. Session reset destroys all content. The structural library grows more precise across sessions without accumulating data.

None of this requires new machine learning research. It requires careful engineering of the components specified in the companion engineering document.

9.2 What the External Version Produces

Beyond demonstrating the architecture, the external version produces the empirical data needed to build the internal version. The stabilisation threshold measured across real content domains. The reactivation signature — what anomaly score distributions look like when infrastructure nodes should be reactivated. The shape of a mature library across diverse domains. The efficiency profile of infrastructure versus observation processing. These are free parameters in the internal version. The external version measures them.

9.3 Timeline Implication

Building the internal version without the external version first means making architectural decisions under uncertainty about all of these simultaneously. With the external version as foundation, each unknown becomes a measured parameter before training code is written. The internal version research timeline compresses from one to three years to six to twelve months. The external version is not a stepping stone to the internal version. It is the argument for why the internal version should exist, and the specification of what it needs to achieve.

THE PATH FORWARD

Build the external version. Validate the twelve predictions. Measure the free parameters. Demonstrate the obedience, efficiency, and novelty detection properties on real tasks. Then build the internal version with empirically grounded targets rather than theoretical guesses.

10. Open Questions

The following questions require empirical work. They are not blockers for building the external version but must be resolved before the internal version can be specified.

Question	Why it matters	Addressed by
What stabilisation threshold correctly predicts durable structural nodes across domains?	Threshold set too low produces premature stabilisation; too high keeps nodes in expensive observation indefinitely	Phase 1 empirical calibration on diverse corpora
What reactivation signature reliably distinguishes anomaly from noise?	False reactivation wastes attentional resources; missed reactivation prevents refinement	Phase 1: inject known anomalies, measure score distributions
How should multiple simultaneous speculations interact? Current formulation sums SEM deltas additively	Competing speculations may produce unpredictable interaction effects	Phase 1: multi-speculation stress tests
What DIVERSITY_THRESHOLD correctly separates generalisable from entity-specific structural nodes?	Threshold determines whether obedience property holds	Phase 1: Wasserstein distance calibration across domain pairs
Can SEM extend to speculations about node identity rather than edge conductivity?	Some speculations challenge whether two nodes should be the same node	v2.0 architectural extension
What is the correct model for multi-entity servants?	Structural refinement from entity A must not encode entity-A-specific patterns that influence entity B	Phase 2: controlled multi-entity experiment

References

Baddeley, A. (2000). The episodic buffer: A new component of working memory? Trends in Cognitive Sciences, 4(11), 417–423.

Bartlett, F. C. (1932). Remembering: A study in experimental and social psychology. Cambridge University Press.

Edge, D., Trinh, H., Cheng, N., et al. (2024). From local to global: A graph RAG approach to query-focused summarization. arXiv:2404.16130.

Graves, A., Wayne, G., Reynolds, M., et al. (2016). Hybrid computing using a neural network with dynamic external memory. Nature, 538(7626), 471–476.

Kurby, C. A., & Zacks, J. M. (2008). Segmentation in the perception and memory of events. Trends in Cognitive Sciences, 12(2), 72–79.

Lewis, P., Perez, E., Piktus, A., et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. NeurIPS 33, 9459–9474.

Lichtenberg, G. C. (1777). De nova methodo naturam ac motum fluidi electrici investigandi. Novi Commentarii Societatis Regiae Scientiarum Gottingensis.

Rumelhart, D. E. (1980). Schemata: The building blocks of cognition. In R. J. Spiro et al. (Eds.), Theoretical Issues in Reading Comprehension. Erlbaum.

Schacter, D. L., Addis, D. R., Hassabis, D., et al. (2012). The future of memory: Remembering, imagining, and the brain. Neuron, 76(4), 677–694.

Tulving, E. (1985). Memory and consciousness. Canadian Psychology, 26(1), 1–12.

Villani, C. (2008). Optimal transport: Old and new. Springer.